# Algorithm Selection for Real-Time Waste Detection

The machine learning algorithms are selected in our research based on the following criteria:

1. **Real-Time Detection**: The algorithm must be capable of detecting and classifying waste in real-time as it is placed inside kerbside bins, ensuring immediate feedback.
2. **Speed**: The model should have fast inference times to process both video input batches and livestreams, providing quick feedback to users.
3. **Accuracy**: High accuracy is essential to correctly classify waste materials and identify contaminants, reducing the risk of misclassification.
4. **Scalability**: The algorithm should be scalable and capable of handling varying workloads, such as multiple video inputs generated by motion detection.
5. **Resource Efficiency**: The model should be efficient enough to run on edge devices or cloud systems, balancing computational demands with accuracy.
6. **Suitability for Video Input**: The algorithm must perform well with continuous video streams, not just static images, ensuring consistent performance across frames.

The above criteria’s ensure that the chosen algorithm can provide accurate, real-time waste detection, with fast and accurate feedback to reduce contamination in kerbside bins.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Algorithm** | **Accepted (OR)**  **Rejected** | **Evaluation Criteria** | **Reason for Acceptance or Rejection** |
| 1 | **YOLOv7**: You Only Look Once version 7 | Accepted | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Optimized for real-time detection with a great balance of speed and accuracy, suitable for real-time feedback on waste classification. |
| 2 | **YOLOv8**: You Only Look Once version 8 | Accepted | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: Very High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Offers better accuracy than YOLOv7 with high inference speeds, ideal for livestream and batch video processing. |
| 3 | **YOLOv9**: You Only Look Once version 9 | Accepted | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: Very High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Newer version with improved efficiency for real-time, complex scenarios using video input. |
| 4 | **YOLOv5**: You Only Look Once version 5 | Accepted | **Real-Time Detection**: Yes, **Speed**: Very High,  **Accuracy**: High,  **Scalability**: High,  **Resource Efficiency**: Very High, **Suitability for Video Input**: High | Lightweight, fast, and accurate, suitable for real-time video processing and immediate feedback. |
| 5 | **EfficientDet-D2**: Efficient Detection version D2 | Rejected | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Removed in favour of better-performing EfficientDet-D3 and D4. |
| 6 | **EfficientDet-D3**: Efficient Detection version D3 | Accepted | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Scalable, performs well for cloud-based training and real-time video detection tasks. |
| 7 | **EfficientDet-D4**: Efficient Detection version D4 | Accepted | **Real-Time Detection**: Yes, **Speed**: Moderate,  **Accuracy**: Very High,  **Scalability**: High,  **Resource Efficiency**: Moderate, **Suitability for Video Input**: High | Higher accuracy than D3 but still fast enough for real-time processing, ideal for detecting contamination. |
| 8 | **EfficientDet-Lite3**: Efficient Detection Lite version 3 | Accepted | **Real-Time Detection**: Yes, **Speed**: Very High,  **Accuracy**: High,  **Scalability**: High,  **Resource Efficiency**: Very High, **Suitability for Video Input**: High | Optimized for mobile and edge devices, suitable for fast real-time detection and feedback. |
| 9 | **EfficientDet-Lite4**: Efficient Detection Lite version 4 | Accepted | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: Very High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Similar to Lite3 but offers higher accuracy, making it ideal for real-time feedback during waste disposal. |
| 10 | **PP-YOLOv2**: Paddle Paddle You Only Look Once version 2 | Rejected | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | The use of YOLOv5, YOLOv7, YOLOv8 and YOLOv9 makes this algorithm redundant. |
| 11 | **EfficientDet-Lite2**: Efficient Detection Lite version 2 | Rejected | **Real-Time Detection**: Yes, **Speed**: Very High,  **Accuracy**: Moderate,  **Scalability**: High,  **Resource Efficiency**: Very High, **Suitability for Video Input**: High | Removed as Lite3 and Lite4 offer better performance and accuracy. |
| 12 | **CenterNet**: Center-based Object Detection Network | Accepted | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Fast object detection, performs well in tracking objects in real-time video feeds. |
| 13 | **RetinaNet**: Retina Object Detection Network | Accepted | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: High,  **Scalability**: Moderate,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Effective in handling class imbalances, suitable for real-time detection with video input. |
| 14 | **NAS-FPN**: Neural Architecture Search with Feature Pyramid Networks | Accepted | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: High,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Balances speed and accuracy for real-time video detection, suitable for waste placement detection. |
| 15 | **YOLOv11**: You Only Look Once version 11 | Rejected | **Real-Time Detection**: Yes, **Speed**: Moderate,  **Accuracy**: Very High,  **Scalability**: High,  **Resource Efficiency**: Moderate, **Suitability for Video Input**: High | Requires more computational power than previous YOLO versions, making it less suitable for ultra-fast real-time tasks. |
| 16 | **Tiny YOLO**: Tiny You Only Look Once | Rejected | **Real-Time Detection**: Yes, **Speed**: Very High,  **Accuracy**: Low,  **Scalability**: Low,  **Resource Efficiency**: Very High, **Suitability for Video Input**: High | Extremely fast but sacrifices too much accuracy, unsuitable for detecting multiple waste types in real-time. |
| 17 | **Faster R-CNN**: Faster Region-based Convolutional Neural Network | Rejected | **Real-Time Detection**: No, **Speed**: Low,  **Accuracy**: Very High,  **Scalability**: Low,  **Resource Efficiency**: Low, **Suitability for Video Input**: Moderate | Highly accurate but too slow for real-time video processing, better suited for offline tasks. |
| 18 | **Swin Transformer**: Shifted Window Transformer | Rejected | **Real-Time Detection**: No, **Speed**: Low,  **Accuracy**: Very High,  **Scalability**: Low,  **Resource Efficiency**: Low, **Suitability for Video Input**: Moderate | While accurate, too slow for real-time video detection, particularly for tasks requiring immediate feedback. |
| 19 | **DETR**: Detection Transformer | Rejected | **Real-Time Detection**: No, **Speed**: Low,  **Accuracy**: Very High,  **Scalability**: Low,  **Resource Efficiency**: Low, **Suitability for Video Input**: Moderate | Accurate but slow due to transformer architecture, unsuitable for real-time detection in video streams. |
| 20 | **TANSformer**: Temporal Action Networks Transformer | Rejected | **Real-Time Detection**: No, **Speed**: Low,  **Accuracy**: Moderate,  **Scalability**: Low,  **Resource Efficiency**: Low, **Suitability for Video Input**: Low | Designed for temporal action detection, not object detection, making it unsuitable for real-time waste detection. |
| 21 | **EfficientDet-D0**: Efficient Detection version D0 | Rejected | **Real-Time Detection**: Yes, **Speed**: Very High,  **Accuracy**: Low,  **Scalability**: Moderate,  **Resource Efficiency**: Very High, **Suitability for Video Input**: Moderate | Extremely fast but lacks sufficient accuracy for the real-time detection of waste. |
| 22 | **EfficientDet-D1**: Efficient Detection version D1 | Rejected | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: Moderate,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: Moderate | Lacks the accuracy needed for real-time waste detection tasks, better alternatives exist in D3/D4. |
| 23 | **EfficientDet-D5 to D7**: Efficient Detection version D5 to D7 | Rejected | **Real-Time Detection**: No, **Speed**: Low,  **Accuracy**: Very High,  **Scalability**: Low,  **Resource Efficiency**: Low, **Suitability for Video Input**: Moderate | Although highly accurate, these models are too slow for real-time video detection tasks. |
| 24 | **SSD**: Single Shot Multibox Detector | Rejected | **Real-Time Detection**: Yes, **Speed**: High,  **Accuracy**: Low,  **Scalability**: High,  **Resource Efficiency**: High, **Suitability for Video Input**: High | Fast but not accurate enough for real-time waste detection, other models like YOLOv7/8 and EfficientDet perform better. |

# Labelling Requirements for Selected Algorithms

## Labelling Format Explanations:

**YOLO (.txt)**: Each object in the image is annotated in a .txt file. The format for each line is: <object-class> <x\_center> <y\_center> <width> <height>, where all coordinates are normalized by image width and height.

**COCO (JSON)**: Labels are stored in a .json file format. Each object is defined by a category\_id (class), and the bounding box is described by x, y, width, and height values. This format is commonly used in the COCO dataset.

**Pascal VOC (XML)**: Annotations are stored in XML files, where each image has details about its objects, including class and bounding box coordinates in the form of (xmin, ymin, xmax, ymax).

## Labelling Requirements for Selected Algorithms:

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Algorithm** | **Labelling Format** |
| 1 | **YOLOv7** | YOLO (.txt) |
| 2 | **YOLOv8** | YOLO (.txt) |
| 3 | **YOLOv9** | YOLO (.txt) |
| 4 | **YOLOv5** | YOLO (.txt) |
| 5 | **EfficientDet-D3** | COCO (JSON) |
| 6 | **EfficientDet-D4** | COCO (JSON) |
| 7 | **EfficientDet-Lite3** | COCO (JSON) |
| 8 | **EfficientDet-Lite4** | COCO (JSON) |
| 9 | **CenterNet** | COCO (JSON) |
| 10 | **RetinaNet** | COCO (JSON) |
| 11 | **NAS-FPN** | COCO (JSON) |

# Creation of the Final Dataset

The list of acceptable items for kerbside bins in Australia has been created using information from two key resources: [Recycle Mate](https://recyclemate.com.au/), developed by the Australian Council of Recycling, and [Recycling Near You](https://recyclingnearyou.com.au/). These websites provide location-specific guidance, allowing users to check the acceptance of items by postcode. Both resources are highly effective in Tasmania, helping residents ensure correct waste disposal and recycling practices.

## Combining Datasets

To effectively train the machine learning model, I combined three distinct datasets that focus on waste classification: the **Classification Model for Waste Materials in Residential Areas Dataset**, the **Waste Segregation Dataset**, and the **Zero Waste Dataset**. The process began by merging the data from the three datasets. Each dataset contained its own distinct set of classes, with overlaps. To consolidate the classes and avoid duplications, I mapped similar classes between datasets and created a unified class list, as shown in the table below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial No** | **Dataset** | **Train** | **Test** | **Valid** | **Classes** |
| 1 | Classification Model for Waste Materials in Residential Areas Dataset | 13,071 | 904 | 1,964 | ['battery', 'bottle', 'cardboard', 'face\_mask', 'food\_leftover', 'food\_peeling', 'gadget', 'glove', 'paper', 'soft\_plastic', 'tetra\_pack', 'tin\_can'] |
| 2 | Waste Segregation Dataset | 6,625 | 775 | 10 | ['Light bulb', 'battery', 'clothes', 'e-waste', 'glass', 'metal', 'organic', 'paper', 'plastic'] |
| 3 | Zero Waste Dataset | 7,030 | 856 | 859 | ['Cardboard', 'Glass', 'Metal', 'Organic', 'Paper', 'Plastic'] |
| **Total** | **Final Dataset** | 26,726 | 2,535 | 2,833 | ['Battery', 'Bottle', 'Cardboard', 'Face Mask', 'Food Leftover', 'Food Peeling', 'E-waste', 'Glove', 'Paper', 'Soft Plastic', 'Tetra Pack', 'Metal', 'Light Bulb', 'Clothes', 'Glass', 'Organics', 'Plastic'] |

## Class Mapping to Waste Categories

The classes in the combined dataset are mapped to the waste categories defined for kerbside bins, ensuring proper classification according to their disposal methods.

* **Landfill**: ['Face Mask', 'Glove', 'Soft Plastic', 'Tetra Pack', 'Clothes']
* **Recyclable**: ['Bottle', 'Cardboard', 'Paper', 'Metal', 'Glass', 'Plastic']
* **Organics**: ['Food Leftover', 'Food Peeling', 'Organics']
* **Contaminants**: ['Battery', 'E-waste', 'Light bulb']

The **landfill**, **recyclable**, and **organics** categories correspond to the respective kerbside bins (general waste, recycling, and green waste). The **contaminants** category includes items that should not be placed in any of these bins, as they require special disposal methods.

# YOLOv5

The below is the execution plan for hyperparameter training of yolov5 algorithm.